LOADSTAR LETTER #47

Arkanix Labs Announces Completion of First Part of "New Plan" Software and Hardware

Seattle (July 1, 1997) -- Arkanix Labs today completed work on their C128 based MOD player known as MODplay 128. This software enables C128 users to play 4 track MODs in four different formats: ProTracker, Sound Tracker, Star Trekker, and Noise Tracker. MODplay 128 can play MODs at 4-bit or 8-bit resolution achieving up to 13-Khz output. The software includes support of REUs up to 2 MB in size for larger MODs.

Also announced were three audio-based hardware enhancements. The first is DigiMAX, a four-channel digital/analog converter. Secondly, is the Dual SID board which gives the C64/C128 stereo sound. Two versions of the board are available to accommodate for the two different SID chip versions.
Finally come the 8-bit Sound Sampler (8BSS), and 8-bit stereo sampler for C64/C128. Included with the package is a free copy of Sound Studio 128 v3.8, sample recording and editing software.

Arkanix Labs is taking a new approach to the selling of their hardware products. The full parts lists and schematics are available from their homepage, this way ordering from them is not required if you can handle a soldering iron. But for the non-soldering types out there they provide a service for ordering pre-made boards.

Programming information for the hardware is available upon request. Pricing information is available from the Arkanix Labs home page at www.arkanixlabs.com, or Email to catalog@arkanixlabs.com will provide an online, text only catalog of our current and future products.
Contact: Petar Strinic
petars@arkanixlabs.com

VideoCam Services Announces "The Internet for C64/128 Users, 2nd Edition," New WWW Domain Name

17 June, 1997. Reynella, South Australia. VideoCam Services is pleased to announce that the second edition of "The Internet for Commodore C64/128 Users" has been published, and has been shipping since the 13th of June 1997. There were several reasons for publishing a 2nd Edition:

The weight of the first edition varied between 498 and 503 grams. Shipping costs rise significantly at 500 grams. Minor typesetting changes were made to the index and glossary to reduce the weight of the book without sacrificing content. Actual book content has increased.

Only one Australian Internet provider was listed in the Appendix. This oversight (as well as minor typographical errors) has been corrected.

Jim Brain of Brain Innovations, Inc. recently announced changes in services offered to the Commodore community, adding FTPmail and Listserv services to replace his MAILSERV program. Additionally, the Commodore FTP site used extensively for examples in the book ceased to exist and was transferred to Jim Brain's site. VideoCam Services felt there was enough change to warrant updating this information in The Internet for Commodore C64/128 Users.

The opportunity to have the book catalogued for the National Library of Australia (similar to the US Library of Congress). This service provides information about books to the worldwide database for books in print and is used by bookstores and libraries for ordering.
The Internet for Commodore C64/128 Users, 2nd Edition by Gaelyne R. Gasson
Published by VideoCam Services
ISBN: 0-646-32207-9

Price per book:	Aust	USA	Canadian
	$36.95	$29.95	$39.80
Shipping:	|	|	|
Intl Express	$15.00	$12.00	$16.20
Airmail	$11.50	$ 9.00	$12.50
Economy Air	$ 9.00	$ 7.00	$10.00
In Australia	$ 5.00	n/a	n/a

Orders can be accepted via phone, fax, postal mail, Email or the World Wide Web. VideoCam Services accepts personal checks, Visa, Master Card, Bankcard and American Express.
New Domain Name

VideoCam Services announced today that the World Wide Web pages formerly located at

http://hal9000.net.au/~moranec

or

http://hal9000.net.au/~rgasson

are now accessible at their new domain: videocam.net.au.

The following World Wide Web pages have been relocated within the Website: VideoCam Services: http://videocam.net.au/

The Internet for Commodore Users: http://videocam.net.au/tifcu.html

Book Orders: http://videocam.net.au/bookord.html

QWKRR128 and Browser: http://videocam.net.au/qtoc.html

Contents Of This Issue

jeff@LOADSTAR.com

wookie@inconnect.com

US MAIL: ATTN. Jeff Jones
J & F Publishing P.O. Box 30008, Shreveport, LA 71130-0008, Phone: 318/221-8718, Fax: 318/221-8870, BBS: 318/425-4382

Gaelyne Gasson's personal and Commodore related pages have changed from http://hal9000.net.au/~moranec to http://videocam.net.au/~gaelyne.

The Commodore FTP site sponsored by VideoCam Services remains unchanged. The directories are located at hal9000.net.au /pub/cbm.

The Adventurer's Blueprint

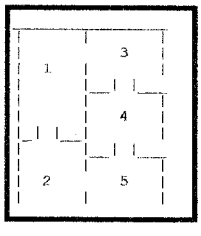
By Scott Eggleston. This past month I have been feverishly consumed with a program I plan to submit to LOADSTAR. This is an adventure game with a souped-up interface, courtesy of LS's toolboxes, and some goodies I've always wanted in these types of games.

Before you can do anything in an adventure, you must know where you want to explore by creating a map. Then, you need to somehow slap this info into your machine so your program can access it.

Many years ago, I bought a book called "Creating Adventure Games on Your Computer" by Tim Hartnell. It covered the basics of this topic, offered in generic BASIC listings so any computer could use it. While some parts are horribly dated, there is a valuable chapter that helped me in starting my game.

This is chapter four entitled, "Creating the Floor Plan." The first thing you need to do is draw a simple map (see illustration). Then, from this map you create a "travel table", as seen below:

Room	N	S	E	W
1	0	2	3	0
2	1	0	5	0
3	0	4	0	1
4	3	5	0	0
5	4	0	0	2



See how that works? If you are in room one, you can only move two directions, south and east. If you move south, you end up in room two, east puts you in room three, and so on for the rest of the rooms. Of course, you can add as any extra directions as you want (such as up and down), you'll just have to expand your table.

To get the computer to recognize these numbers, you need to first change your table into data statements:

```
100 data0,2,3,0
110 data1,0,5,0
120 data0,4,0,1
130 data3,5,0,0
140 data4,0,0,2
```

Next, you need to read them using one of two methods. The first involves using a two-dimensional array, and can be implemented by the following addition to your program:

```
20 forx=1to5
30 fory=1to4
40 readro(b,c)
50 next:next
```

These lines could of course be combined, but have been shortened for best presentation in this newsletter.

Line 20 begins the loop, 5 being the numbers of rooms to be read in. Line 30 is the number of directions possible from each room.

If you have more than 10 rooms, or 10 possible directions, you'll have to use a DIMension command. If TR=total rooms, and PD=possible directions then the DIM command would look like this:

```
10 dimro(TR,PD)
```

When your program is run, the computer will see each "room" like so:

```
ro(1,1)=0
ro(1,2)=2
ro(1,3)=3
ro(1,4)=0
ro(2,1)=1
ro(2,2)=0
.
.
.
ro(5,4)=2
```

This is the "array way." It is a little easier to implement, but is also a memory sponge, taking up valuable space in your BASIC program area. Due to this, I prefer the second way to access data, by POKEing the room values into an available memory slot, outside of BASIC. This technique is excellently detailed in LOADSTAR #155 by Knees Calhoun.

The idea is to POKE the values contained in the data statements, then PEEK them when needed. In my program I put this data starting at 23296. The following lines would accomplish this:

```
5 ro=23295
20 forx=1to20
30 ready
40 pokero+x,y
50 next
```

The computer will now see the values in a sequence, so when PEEKed the following values will read:

```
23296=0
23297=2
23298=3
23299=0

23300=1
23301=0
.
.
.
23316=2
```

Please note that line 30 reads as READ Y, and not the word "ready" Also remember that DIM will again be needed if your total rooms or possible directions exceed 10.

This is the best method in my book. It saves on variable space, and the whole block of memory can be BSAVEd, if you want to save changing room configurations (such as if you discover a secret exit to a room) in the middle of your game. Another nicety is that this info can be put anywhere in available memory by changing the value of RO in line 5.

To use this information in your program, you must first let the program know what room you will begin in. Let's say you are in room 1, so set your room number variable like so: RN=1.

Next, you'll want to set up a loop to tell you what directions of travel are available. When using an array, your loop could look like this:

```
200 ifro(rn,1)<>0
    thenprint"north"
210 ifro(rn,2)<>0
    thenprint"south"
220 ifro(rn,3)<>0
    thenprint"east"
230 ifro(rn,4)<>0
    thenprint"west"
```

For the POKEd version:

```
200 ifpeek(ro+(rn*4-4)+1)<>0
    thenprint"north"
210 ifpeek(ro+(rn*4-4)+2)<>0
    thenprint"south"
220 ifpeek(ro+(rn*4-4)+3)<>0
    thenprint"east"
230 ifpeek(ro+(rn*4-4)+4)<>0
    thenprint"west"
```

This is about as simple as you can get. It's also nice to create a loop to do the work of all four lines, but that will require you to read the directions into a string:

```
60 forx=1to4
70 readdi$(x)
80 next
150 datanorth,south,
east,west
```

Now the POKEd version can look like this:

```
200 forx=1to4
210 ifpeek(ro+(rn*4-4)+x)<>0
    thenprintdi$(x)
220 next
```

For the array, change line 210 to:

```
210 ifro(rn,x)<>0then
printdi$(x)
```

To travel between rooms, you'll have to have an interface. I'm using a point-and-click version with the aid of LOADSTAR's Mr. Mouse toolbox, and sprites defined to look like buttons. You can use any interface, just slap the desired direction number into a variable (such as DI), and run it through a formula to tell you what new room you end up in.

For example, if you want to go north, DI=1, and you could gosub to the following routine to check it:

Array version

```
1000 ifro(rn,di)=0thenreturn
1010 rn=ro(rn,di):return
```

POKEd version

```
1000 ifpeek(ro+(rn*4-4)+di)=0
    thenreturn
1010 rn=peek(ro+(rn=4-4)+di:
    return
```

Line 1000 sends you back, since you cannot walk through walls. Line 1010 will send you to the new room.

I realize this is all very simple, but before I got the aforementioned book, I had no clue how to set up an adventure map. It's up to you to flesh out these bare guidelines, adding your own distinctive style to your game. Adventure games can be a lot of fun, both to program, and to play. I hope this article will encourage some of both.

A Cure for Internet Spamming

By Jeff Jones. Email is free. That can be a bad thing because Email marketers don't care if they waste a letter on an extra 250,000 people who are not potential customers. Sure, the shotgun approach works sometimes. Send out a million letters and maybe a half a percent of them will respond. Send out two million and perhaps double your return. To the untrained and possibly uncouth marketer, this seems like good math. It isn't. There's also a science behind marketing. There's a very good reason why you receive paper junk mail every day and probably don't mind.

Ironically, I received the following Email while I was writing this article:

LET US DO YOUR BULK MAILINGS...$200 PER MILLION!!!!

Our company will do bulk emailing for your product/service. Addresses are extracted daily by two of our computers, which run 24 hours a day 7 days a week, scanning the net for new addresses. They are fresh! Over 35 million addresses on file.

Our prices start at $200 per 1 million mailed. No more than 2 pages (50 lines), no porn and no foul language. We do not do targeted mailings at this price.
There are no lower prices on the net. All mailings can be done in a matter of hours. XXX has 6 computers sending messages daily.

For the fastest service, cheapest prices and cleanest mailings call xxx-xxx-xxxx. If line is busy, just keep trying, as we are very busy.

Now when you think about this, the company is adding names to its mailing list in an invasive way. They probably got my address because it appears on the LOADSTAR web page. They probably get most names from newsgroups.

I recently Emailed sales@earthlink.com because I dread getting any Email from anyone at earthlink.com. As a matter of fact, I actively delete any mail from this domain that slips past my defenses. When I found that Earthlink appeared to be a legitimate Internet service provider, with legitimate customers who might even be LOADSTAR subscribers, I had to Email them and pressure them to place their rogue customers on a leash. What follows is the body of my letter to them. Hopefully they will take me seriously. Softdisk Internet Services simply doesn't allow spamming, which I applaud. If Softdisk can do it and stay in business, so can they.

Dear Earthlink,

I just realized I get an enormous amount of spam from your domain. Is there a way you guys can start charging people extra when they send enormous volumes of Email? I sincerely believe that if spammers had to pay half as much as normal businesses do for bulk mail, they would *target* their potential customers.

I don't mind my paper junk mail because it's *targeted*. I don't mind some of my electronic junk mail because it comes from legitimate companies I've done business with. Since I don't believe I can earn $800 per week doing things like stuffing envelopes, a lot of the unwanted Email I receive insults my intelligence. Requests for them to stop Emailing me might as well fall on deaf ears when they turn around and purchase my name again in another list.

If ISPs don't do something to curb spam, that value of Email will continue to decrease. Also mailing lists, which you might one day wish to sell, and ad space, which you will also want to sell, will be *devalued*. I take the net seriously. Spammers make it hard for legitimate companies to make money on the net.

You can't respond to me from an address in your domain or "savetrees.com," "earthfriends.com," or "earthstar" if you have anything to do with those domains. I have those domain names *blocked*.

Please stop this annoyance.

.... Jeff Jones

Thanks for the Memory...

By Gaelyne R. Gasson. NOTE: This article was originally slated for BBS Magazine (BBS.Net), but was never published. It was written just prior to the demise of the magazine. It's been updated for the LOADSTAR Letter.

When I was new to computing, I used to browse software manuals and anything else I could get my hands on to learn about the Commodore. I remember reading in one manual about something called an REU, but the author assumed that everyone would know what this is, and what it does. I couldn't glean from the surrounding text exactly what it was, but it sounded like something I might like to use, as it seemed to expand the capabilities of the program and possibly even the computer, too. Since then, I've found the meaning of this acronym (RAM Expansion Unit). I've also learned that an REU can enhance programs that accommodate it. In turn, this expands the Commodore's abilities.

A RAM Expansion Unit (REU for short, pronounced "ARE E YOU") can come in different forms, but basically is a device you plug into the Cartridge port of the computer. As its name implies, it has extra RAM.

RAM is memory that can be used as either program memory space, or as temporary (but speedy) disk space. Commodore marketed three different RAM Expansion units, each with more

memory than its predecessor: the 1700 (128k), 1764 (256k) and the 1750 (512k). Originally, the 1764 REU was for the Commodore 64 only, and the other two models were solely for the C128. However, the C64 can use the other models, providing you use a heavy-duty power supply, which Commodore also sold.

Typically, Commodore stopped marketing the 1750 REU just as they became a hot item, and never did start production on them again. A third party (Chip Level Designs) took it upon themselves to market the 1750 Clone. This is the size of a normal games type cartridge (about 3" long by 2 1/2" wide). The Commodore REU is quite a bit larger than the 1750 Clone, about 5 3/4" long by 4 3/4" wide, not including the section that slides into the cartridge port. There's an advantage to the larger size of the Commodore REU, as they can be expanded to hold more memory. The earlier model REU that I bought used says it's a 1764 on the outside, but it had been upgraded to 512k at some stage. You can download plans that tell you how to do such an upgrade yourself. Look for 1764exp-512k.txt, 1750exp-2MEG.txt, bynd512k.sda and bynd512k.arc. If you're not comfortable with making these changes yourself, Raymond Day (rday@genie.com or Voice 313.699.6727) can upgrade your REU for you - call or send him Email for the latest prices. Until recently, there were no "new" REUs and you had to find them on the used market. Recently, (July 1997) Creative Micro Designs Inc. have begun marketing two new REUs, the CMD 1750 (512K) and CMD 1750XL (2 mb). These sell for $99 and $139 respectively.

... But what do you do with an REU?

An REU sits in the cartridge port and waits for you to load a program that uses it. Or, you can load RAMDOS and use the memory as a RAMdisk. RAMDOS is a small program that wedges itself into the operating system and lets you access the REU. This is like adding another disk drive to the computer, only it's faster and doesn't make any noise. The only drawback is that it's not permanent.

A RAMdisk is like a "scrap disk" where you can save things temporarily. Because this doesn't depend on a mechanical means of storing or accessing the data like a disk drive does, it's very fast. When you turn the computer off though, anything stashed in the REU will be gone. Not all programs can handle the fact that RAMDOS wedges itself into the operating system, but for software that's RAMDOS compatible, it can be quite useful.

At one point in time, using the REU as a RAMdisk was pretty much its only use, but in time programmers have learned to use it in different ways. Some programs (such as Maverick and Big Blue Reader) store data to the REU when copying disks or files, using it as a temporary buffer area. Given an REU with enough memory, Maverick and C-Kit can copy entire disks to another without need to access the original once it's stored the data in the REU. Ipaint and SprayPaint use the REU to store program routines so they can be accessed seamlessly by the user. GEOS uses the REU in two ways. It uses part of the memory for some of its key routines, but it also allows the user to have a GEOS RAMdisk and that greatly improves the speed of the system. QWKie (the C64 off-line mail reader) lets you read mail from the REU, The Write Stuff and PaperClip III uses the REU to store their dictionary files so spell checking is faster. Will using an REU make a difference when BBSing?

What you get out of an REU when phoning BBSs and the Internet depends on the term program and what you're using it for. Term program authors have used the REU in both manners discussed above - either as a RAMdisk or as extra memory for the program/computer/user to use. If you spend much time capturing text online, an REU used as buffer space gives you more room for what you capture, with less frequent saving to a disk drive.

Dialogue128 offers the choice of using RAMDOS so you can have a RAMdisk, or you can use the REU as buffer-capture memory. DesTerm will use the REU as a RAMdisk, but installs its own custom form of RAMDOS, which means you must use DesTerm to copy your downloads to disk when you've finished. Novaterm 9.6 takes the same approach as DesTerm, but also uses a portion of the available memory to control the screen display when online. Versions of Novaterm before v9.6 make use of RAMDOS and automatically create a RAMdisk when it detects an REU as the program loads. All three programs can use the memory for fast access to script files, too. Fritzterm uses the REU as a huge (and very fast) buffer area.

When it comes to downloading, your "kilometerige" (mileage) may vary depending on which term program you use, and what your downloading habits are. If you're using a program that installs RAMDOS, download speed will be a little slower (not much though) than if you use a program that uses a customized RAMDOS and RAMdisk. For those who must phone long distance, downloading to a RAMdisk is faster and will definitely save you time online. However, if you're modeming locally, you need to consider that you'll have to spend time off-line copying your files to disk. If you take this time into consideration too, you may not be saving any "real" time, although you'll spend less online. This applies to uploads too, especially if the file you're uploading isn't very big. For instance I call my local Internet Provider and upload a QWK reply packet to send out my Email. This file usually is under a hundred disk blocks. If I took the time to copy it into the REU first, I'd probably be adding to the time I spend within the term program. On the other hand, if you have several files you want to upload to the BBS, it's worth copying them to the REU and using a batch protocol to upload them.

A last consideration is the use of streaming protocols with high-speed modems. Novaterm 9.6 is currently the only Commodore term program with streaming protocols. Due to the nature of the computer, the only way these protocols can work is if you transfer files to or from the custom RAMdisk. Whether it's worth it or not in personal time depends on the size of the file, and how much need there is to make every minute of your online time count. If you don't mind spending an extra few minutes off-line copying files, and if you have a lot of data to move, then making use of this is logical. However, if you're typically transferring a QWK mail packet and an even smaller reply packet, it may not save you any personal time at all.

A RAMLink is not a replacement for an REU. When Creative Micro Design's Inc. first began marketing the RAMLink, many thought it would act as a replacement for an REU. Considering that the RAMLink can be purchased with varying amounts of memory that's used as disk storage, they have similarities. Unlike the REU however, the RAMLink has a separate power supply, so everything is retained when the computer is powered down. Programs that recognize the REU do not see RAMLink's memory as REU space except under GEOS and other special conditions. Some programs now use RAMLink memory in this capacity, but it isn't the same. The way the computer accesses this memory is different too. The advantage of a RAMLink is that it has a port for the REU, and can hold the contents

(Continued on page 6)

WEB WATCH!

Net Mechanic
Tune Your Site

http://www.netmechanic.com/

If you author web pages, you might want to try an HTML validating service. This is a site that you visit, inform it of your site's location, and then wait for an hour or two while it visits and grades your site. It also checks your site for broken links. I had it check out LOADSTAR's site and it had quite a bit to say. It will likely find errors that you missed. I contacted the creators of Web mechanic and got this quote from them:

"We created the site to promote our abilities to create robots and to provide a service to the web community. We hope to benefit in the future through a fee for service Pro version with more capabilities and a Pro version of the software for hosting on intranets. We are also planning to place advertisements on the free pages to cover the costs but have not had time to finalize those arrangements."

"Mainly we get positive feedback from our users and a lot of repeat traffic. We do however get bug reports which help us to improve the product by making it more resilient. We are quite new and there are still things to improve and correct but in general we seem to have a fairly good loyalty among our users which we greatly appreciate."

Robots are a fact of life on the Internet. If you've used Lycos, Alta Vista, InfoSeek or any other search engine, you've tapped into databases which were generated using robots. Robots are software tools. They are not inherently good or bad. However, poorly written robots can do serious damage to a Web site. Since a robot is an automated program, it can submit HTTP requests to a server at a very, very fast rate. This can overload the server to the point where it can't respond to requests from real human beings using real Web browsers.

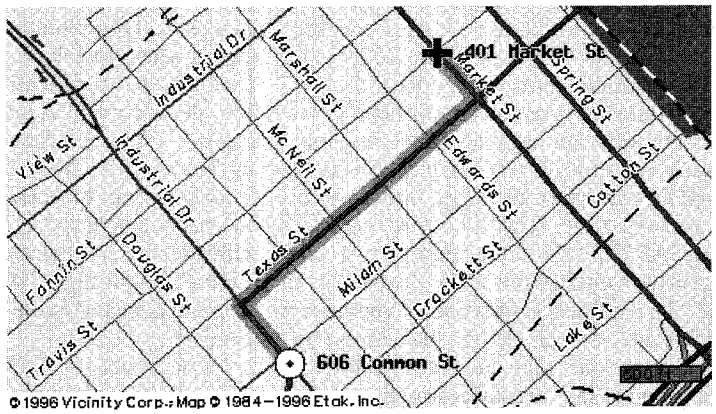
Our robots pause between each access to your site. A Web robot is any automated program which requests HTML files from a server. A robot typically processes these files to find links to other Web pages, then processes these pages too.

The errors spotted by an HTML validator typically fall into two categories: bad tag usage and non-standard extensions. The first of these cases is straightforward. The HTML standards define the minimum level of performance for processing HTML tags. In practice, Netscape Navigator usually exceeds this level and overlooks minor mistakes. Other browsers may not be so kind. In any event, correcting these mistakes is simple and doesn't require much time. So why not fix them?

The second type of error -- non-standard extensions -- is harder to judge. In this type of error, your page uses a tag supported only by Netscape Navigator or Microsoft Internet Explorer. You will have to weigh the value these tags add to your Web page against the number of visitors using incompatible browsers. Professional design firms routinely track the type of browsers hitting their pages, and make just this sort of tradeoff decision. If 98% of your visitors can handle the tag and it adds real value to your page, you may want to keep it.

JEFFREY T. MORGAN, jeff.morgan@montesano.com ,Monte Sano Software, LLC, http://www.netmechanic.com

Yahoo Maps Better than AAA? http://www.yahoo.com/



Driving directions, found in Yahoo Maps, is a neat feature in Yahoo.com — and you don't need a graphical browser to use it. Just surf on over to www.yahoo.com and select *Maps*. Type in most any address in the United States and you get a nice map of the area. Lynx users won't appreciate that, but they will appreciate the next feature. Select Driving directions and then type in another address, and you'll get plain English instructions on how to get from point A to point B. The trip limit is 700 miles. If you don't like the directions you can have it generate an alternate plan.

The plain English instructions print out well, and they work better while you're driving and yelling at your kids. I hope they never get rid of this service because I find myself using it every week. It helped me find my last two weddings and a place of business.

The map above shows you how to get from Harrah's Casino to LOADSTAR. Actually it's easier to just get off of Interstate-20 and head north on Common to 606 Common. But let's say you follow all the signs to Harrah's, lose your money, and then want to stop in and see us. The map above shows the directions. Here are the English directions it gave me:

MILES: 1
STARTING AT 401 MARKET ST, BEGIN ON MARKET ST (US 71, HWY 1) HEADING SOUTHEAST 0.02 MILES. TURN RIGHT ON TEXAS ST (US 79, US 80) HEADING SOUTHWEST FOR 0.5 MILES TO 606 COMMON ST.

Then there's their disclaimer: *(NOTE: like any driving directions/map you should always do a reality check and make sure the roads still exist, watch out for construction, and follow all traffic safety precautions. This is only to be used as an aid in planning.)*

They have a point about the "reality check." Make sure your data is correct. Harrah's Casino is not at 401 Market, but according to the white pages, it is. As for Yahoo's maps, one of them listed Greenwood Road as Greenville Road or something. Note that I tried to find some other Shreveport landmark to use as an example, but no matter how far away they were, the directions ended up being two or three turns and didn't show off the intricate directions.

of the REU even when you turn the computer off. The memory from the REU can be used in addition to RAMLink's memory.

UART Cartridges and Added Memory. In order to use high-speed modems, we need to use a UART cartridge such as the SwiftLink or Turbo-232. Using these in combination with an REU and a RAMLink can give you the ultimate in fast telecommunications. As you've probably discovered though, you can only use the UART cartridge or the REU, but not both at the same time. This is because the REU and UART use the same addresses, so one is always switched out while the other is in use. To resolve the situation, if you have a SwiftLink, you can change the address by opening the cartridge and cutting a trace, according to the original instructions that came with it. CMD is now marketing the Turbo-232 to replace the SwiftLink. Instead of traces that you need to cut, the Turbo-232 has jumpers that can be moved to change the address.

For the RAMLink, CMD offers a replacement chip (P/N:RLDIRECT) that lets you access the both the REU and UART cartridge, no matter how the RAMLink's Normal/Direct switch is set.

The SuperCPU64 and SuperCPU128: CMD's new SuperCPU 20 MHz accelerator for the 64 and the 128 will eventually allow us to use added SIMM memory. The SuperCPU's memory will not be available for disk-type storage however. It'll be used by programs written to take advantage of it and other aspects of the SuperCPU. New killer applications will be coming out to surpass what we already have... I can only imagine what the next generation Commodore BBS software will be like at 20 MHz and an additional meg or two of computer memory!

Gaelyne Gasson is the author of "The Internet for Commodore C64/128 Users". She can be reached via Email (gaelyne@hal9000.net.au), and you can read other articles she's written on her web page at http://videocam.net.au/~gaelyne

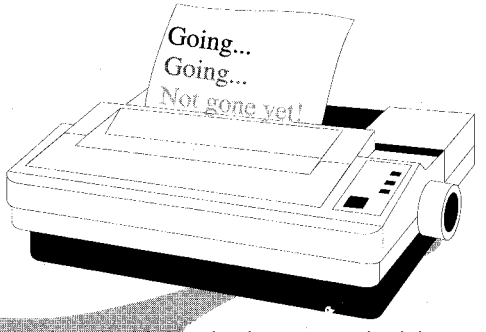
Dot Matrix Versus Ink Jet

By Jeff Jones. As far as I am concerned, last month's LOADSTAR Letter was a disaster. The disaster was all due to my ink jet printer. Will I take any credit for the shabby way it looked? Not at all. It was the ink jet's fault. I own an Epson Stylus II, and it prints all the LOADSTAR Letter pages that don't have pictures on them. When I realize its limits it usually does a decent job. Oh, this page was printed at 3540 dpi on an expensive imagesetter, costing way more than my life is worth. That's again because that's a picture of a printer on this page, and it's in shades of gray. My ink jet can print shades of gray, but because of the way it's *screened*, Complete Custom Printing, who printed this newsletter, can't capture them as they print. This could be because the dots spread when the ink hits the paper and are not uniform. Specifically, they want 133 lines per inch with the dots at a 45 degree angle. Curiously, if I were to print shades on an "ancient," noisy dot matrix printer, the pro printer will have no problem with it because the dots are so coarse. Blur the dots with soaking and there's a problem. A laser printer? No problem. The dots are all nice and neat and make good screens.

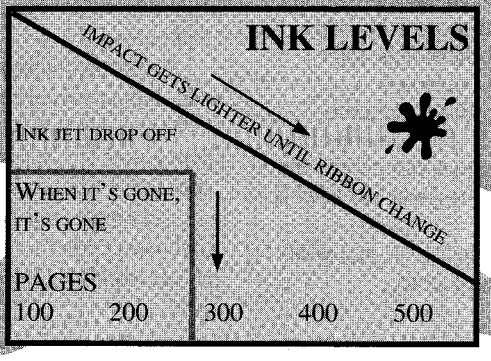
I've used $100 Daisy wheels and $50,000 color lasers. I have to say that the only type of printer that I have a gripe with is the ink jet, which seems to have taken the world by storm. Ink jets are clearly better on paper — well let me rephrase that: Ink jets clearly come with better spec sheets than the venerable impact dot matrix printer. I could be wrong, but you'll not find an impact printer that has a resolution higher than 360 dots per inch (that's DPI to us hardened editors). On a good day, letter quality on my 720 dpi Ink jet printer looks far superior to any 300 dpi laser or 360 dpi impact printer. The Epson Stylus is blazing fast, too. It can spit out a page before you know it. But ink jets have a lot of weaknesses compared with their ancestor, the impact printer:

Inferior Transport Mechanism: Sheet feeders always had their problems, whether it's the copying machine at work, or the laser printer jamming, we just don't seem to have the technology to faithfully have a machine pick up a sheet of paper and guide it through a machine. So if a multi-thousand dollar copying machine, and a multi-hundred dollar laser printer have problems with sheet feeding, what kind of mechanism would you expect to find in a $150-$200 ink jet printer? In a word, *cheap*. I've had a good couple of months, but I still cringe every time the Epson grabs the next sheet of paper. Will it catch or will it try three times and then fail? Will it do like it did last month and actually think it grabbed a page and then spew ink all over itself? After it played out that fiasco, the next twenty pages came out smudged and dirty. Until they get this technology fixed, why not sell tractor feed ink jets? That way people could set narrow margins if they wanted, and print to the bottom of the page. And remembering the old days when we could make banners without having to cut and tape sheets together?

Inconsistent quality: From ream to ream, the quality of print varies with ink jets, even when you purchase so-called ink jet paper. If the paper is too absorbent, the ink just soaks through and spreads, making all your letters appear messy as some did in last month's LOADSTAR Letter. Last month was a month where even in a ream that did well for me, there were patches of bad paper.



An impact printer slowly uses up the ink from its ribbon



Too much blackness in a graphic can give you an outright *soaked* page — so soaked that the page warps and puckers.

Expensive Ink Cartridges: A black ink cartridge costs me $22 from Wal-mart, and color runs me $32. That's expensive, especially when you consider that they have a significantly shorter lifespan than

your average $8 to $10 dollar ribbon for an impact printer. Since I purchased my Epson ink jet last May, I've easily spent $150 in ink — and believe me I don't print except when necessary. By next year I will spend more money in ink than for my printer and paper combined. LOADSTAR's Panasonic on my desk has only been re-cartridged once in *five years*, and since then re-inked with Ebonize spray only once.

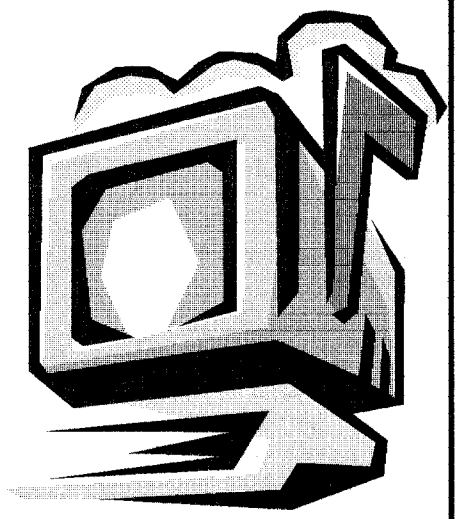
True, in that time, I've had some pretty faint print here and there, but faint print is a lot better than the call to reality you get from an ink jet cartridge when it simply refuses to work anymore. There's no putting up with light print. You *can't* print at all.

Inconsistent paper: Clay-coated ink jet Paper is more expensive, but does produce better results because of its slower absorption rate. However some papers claim to be made for your ink jet printer when it's only plain copying paper. I regularly use laser/copying paper in my ink jet, but from ream to ream it's literally a crap shoot.

I once purchased some Great White paper, labeled as "ink jet" paper, which produced such consistently bad results that I had to give it to my daughters as drawing paper. But what more can you expect for $3.70? Wal-Mart sells more expensive "ink jet" paper that does a better job. Epson's own coated paper costs somewhere over $10 per hundred sheets! I used to use it for LOADSTAR Letter masters until I ran out. Epson glossy color paper (actually it's tough, unrippable plastic) costs over $2.00 per sheet!

I may sound biased toward tractor feed printers by now. Actually I'm not. The next printer I'll buy will be a 600-1200 dpi laser printer. These printers have come way down in price since their inception. Ink jets have come way down, too. It wasn't long ago that Epson's 600-dpi color ink jet printer, inferior to my 720 dpi, $250 printer, was called "professional" and cost over $1200. However ink jet accessories are as high as ever. The ink, which flows like rivers in America, stays at high prices though available everywhere. The printers still soak paper, hurriedly wasting ink so you can go out and buy more ink.

So what's happened to the impact printers? CMD still sells them at reasonable rates, along with the interfaces needed to connect them to your C-64/128. I've noticed that a lot of the old 9-pin printers have been dubbed "professional" office printers now. Since people use them to print form-fed invoices and statements, they are used for business reasons, but I doubt if that



explains why they should cost more now than a lot of the 24-pin printers.

Reader Mail

WHERE'S LS-ZIP?

Dear Jeff,

I have a suggestion for a program that LOADSTAR maybe could benefit from publishing. Recently, I have found that I have encountered a problem. I have downloaded many files from C=64 file areas, that are supposed to work on the C=64. My problem is that I cannot seem to "unzip" the files that end with the .zip suffix.

I have gotten some information from the net regarding this problem. My thought is, how difficult would it be for a programming genius like yourself (a little gratuitous flattery, with the hope of prompting you to arise to the challenge, ☺, to write a program that will "unzip" all such files that end with a ".zip" suffix. Just some thoughts. Thanks for the great work you guys are doing at LoadStar!

Your fellow C=64er,
Bob Dallmann
Sent from Innovations BBS, 32 Lines
Local #: (718) 575-2914 300-33.6k

Jeff: A genius I may be on some planet, but compression is something I've never looked into for more than a few hours. I understand how it's done, but have never even seen the code needed to do zip compression. I'm the wrong guy to ask.

WHERE'S THE LS CD-ROM?

Dear Jeff

Have you guys ever thought of making all issues of LOADSTAR 64/128 available on a CD-ROM with a 64/128 emulator?

Just a thought...

Stephan Rohatynsky
steroh@aahz.magic.mb.ca

Jeff: We had a small informal meeting about this and decided that it was possibly worth looking into. The problem is that the only reliable emulator we'd want to use would be the C-64S emulator, which would push the price of the software up by thirty or forty dollars. Sure, all of LOADSTAR and all LOADSTAR products, could fit on one CD-ROM. The problem is, how much do you charge for it? $100. I asked one person and he said $50. Now imagine me using geoBEAP or any other known admittedly slow conversion utility to create over 500 .D64 files and then transfer them to a PC. It would take, days or even weeks of pure disk swapping before I even begin writing the CD. At what point does the profit overtake my colossal salary? We'll discuss it more, but since we can't see the emulator community forking over much money, we can't see ourselves devoting much time to a product that only ten people will buy.

LITTLE RED BLUES

Dear Jeff:

HELP! I am trying to learn how to run Little Red Reader-128 which is on LOADSTAR Extra#1.

My equipment is as follows:

- COMMODORE 128 WITH JIFFYDOS
- COMMODORE 1902 MONITOR
- COMMODORE 1581 WITH JIFFYDOS
- COMMODORE 1571, PLAIN
- COMMODORE 1541 WITH JIFFYDOS.
- I HAVE ONLY 1581 AND 1571 PLUGGED IN WHILE TRYING TO RUN.

Symptoms: Program loads OK in either 40 or 80 col. mode. First thing I try to do is read MS-DOS directory. I get following error message:

MS-DOS disk error #9
27,read error,27,28
or
27,read error,00,01

This error occurs on all my MS-DOS disks, even the commercial ones which I have not written to. I have turned off both disk drives, reloaded and tried. I have turned off entire system, reloaded and tried. It will simply not read the MS-DOS directory for some reason. I have tried in both 40 and 80 column mode. I do not know anything else to do. Have you any suggestions?

I will be editor of MaiLink the newsletter of Meeting 64/128 Users Through the Mail for September and expect to get some submissions via e-mail and would like to convert them from 1581 to 1571 and subsequently to TWS format. The program Little Red Reader-128 is on a 3 1/2" disk which my 1581 reads ok. I cannot think of anything else that might help pinpoint the problem. If there is, please let me know. In the meantime, HELP!

Walter.

Jeff: The first question I can think of is: "What type of MS-DOS disk are you trying to read?" These days it's so rare to find a low-density 3.5-inch disk, especially a commercial one. Little Red Reader does support high-density disks, but only if you read them with an FD-4000/2000. The Commodore 1581 is simply not a high density drive. It can't handle high-density disks, sometimes even when formatted as low density. They may read a high-density directory, but will not faithfully read data from a high density MS-DOS disk. I would suggest copying the MS-DOS data to a properly formatted 720K disk and then trying Little Red Reader. If this doesn't work, check your 1581 drive. It might be defective.

SPRITE CAPTURING COMMODORE SIDE?

Usenet Feedback:
xy3951@epix.net wrote in article <5pb9vonve4@news1.epix.net>...

I was wondering, what is the best way to get a perfect screen capture? I have a Super Snapshot v5 but it won't capture sprites, though it will print them out with the screen. I would like to do a capture so I can convert and put the picture directly in my newsletter. Any suggestions, or is the only way to really do it is to take a picture and paste that in the newsletter then copy it? And if that is the only way, any suggestions on the best way to take a picture of the monitor?

Thanks in advance.
Dan Barber
xy3951@epix.net
burglar@replay.com

Jeff: Snapshot doesn't attempt to capture sprites because sprites aren't part of the screen. They are overlaid. Also, you can have a high-res text or bitmap screen and a multicolor sprite with entirely different resolution and color rules. This would be impossible to re-display on a C-64 without programming new sprites -- which (duh) actually wouldn't be hard to pull off if a new page standard were devised that re-loaded the sprites and any raster code needed to display them.

Anyway no file format exists that handles this extra information though you wouldn't have a problem copying the screen and sprite data to a BMP or GIF file -- if you are a decent programmer. I've toyed with writing a Window's BMP converter, but I don't think it would be very appreciated Commodore-side.

Personally I use my C-64S emulator to capture sprites and fancy screens. It captures sprites fine. Curiously my Amiga Emulator by Questronix fails to capture sprites even though its sprites (at least four of them) are software sprites because the Amiga only has four hardware sprites.

UNLIKELY TRADE OFFER

From bj1nx@voicenet.com
(D Murray)

Looking to trade my Amiga 2000 setup for a Commodore 128 setup, or huge 64 package must include the following:

- HARD DRIVE
- ACTION REPLAY CARTRIDGE V6
- RAMLINK OR FD FLOPPY,
- AND WHATEVER ELSE!

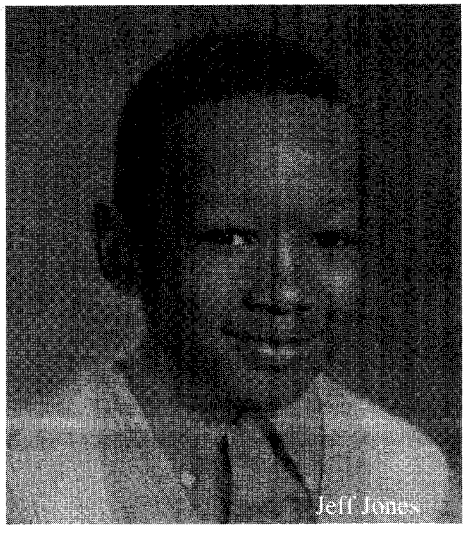
Amiga 2000 setup is as follows

- A2000 WITH 3.1 ROMS (ALSO COME WITH SWITCHER AND 1.3/2.1)
- 68030/25 WITH 8-MEGS 32-BIT RAM
- 1 GIG SCSI HD
- DUAL FLOPPY
- FLICKER FIXER
- AND MORE

Reason for trade, the Amiga doesn't EMU commie 100% and that's all I'm

using it for (any other apps I use the IBM).

Jeff: This goes to show how fun a Commodore 64/128 can be. I too own an Amiga system faster than the one mentioned above. I use it almost exclusively for MIDI work. Other than that, I simply don't turn it on. The Amiga-based Commodore emulator is slick and smart, but slow and incompatible. The PC emulator is fast and not slick, but very compatible with even the most timing-sensitive demo code.



To Print Or Not To Print?

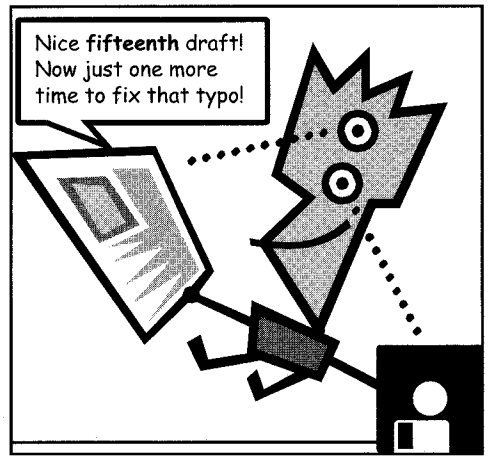
An editorial by Jeff Jones, Shreveport's youngest professional Commodore programmer. People like to print. Some people don't care what they print as long as they print — because they consider it closure of the computing process. After all, if you don't have a hard copy to show for your hard work, what's the point, right? Wrong!

Printing kills more trees than any fungus, beaver or logger. I make an effort to print only when I have to peruse a sheet for errors, and even I throw away unbelievable amounts of paper. Curiously I've never had the printing bug, but I receive a lot of mail from people who tell me that they actually enjoy printing. After moving over to ink jet, I see printing as an expense, and actively avoid it. It's not only a money thing, it's a resource thing — and printing something invites me to promptly lose it.

At the LOADSTAR Tower we produce a huge amount of text. Sometimes it blows my mind how much text we can pack into an issue. I couldn't fathom printing it all yet if there is one file that won't print under Text Printer, we'll hear about it. That's because we decadent Americans love to print. Some of our readers tell us that they print out every single article because they prefer to read it on paper. That's actually a good reason to print. The flip side of this notion is that you probably should avoid printing a file that you will never read again. I'm glancing to my left, toward my garbage can, at the paper sticking up. Hey, I'm guilty! I'm not as guilty as most, but still I waste tons of paper!

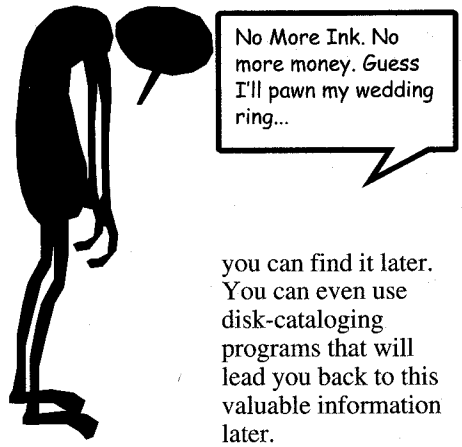
I would advise against printing every article in LOADSTAR just for the sake of doing so. This not only costs you money in ink and paper, but also will ultimately cost you your printer. Take a look at your spec sheet! Your printer actually *tells* you that it will print an average of so many million characters before it rolls over and dies. I don't like hurrying my hardware toward the repairman.

Once I came across a nice, concise, 1750 REU article on the Internet, and printed it out. Now I have no idea where it is. I could have saved the article as a file in my programming directory and never lost it. I tried to explain this to a friend of mine whom I attempted to navigate through the Internet while on the phone. She came across an interesting page and asked me to hold on while she printed it. I told her then that she should have probably saved it. She told me that she would probably lose the sheet or never read it. She then began to complain of the cost of ink, with which I commiserated.

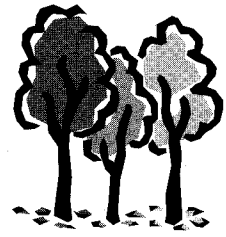


Printing should be delayed until proofing time

Saving is a fantastic alternative to printing. Saving a file requires that you go through a filing process. You'll probably place the file on a labeled disk or a properly named subdirectory. What's more, you'll give it a proper name so that you can find it later. You can even use disk-cataloging programs that will lead you back to this valuable information later.



I confess that most of the stuff I save off of the Internet is falling to bit rot on my hard drive. I'm probably never going to read it again — but at least I didn't spend money printing it.



Okay, we're all allowed a honeymoon. If you buy a new printer, have a print fest! Print out banners and signs and all of two issues of LOADSTAR. Now when you're headed back out to office depot and looking in catalogs for the cheapest prices on re-inking utilities, you must tell yourself that the honeymoon is over, if not for yourself, then for the trees.

LOADSTAR Library Bug Finally Revealed

Bug and text by Jeff Jones. A number of people have complained that LOADSTAR Library 1-100 will not run. This is a simple fix. Just copy *library 1-100* to a separate disk, away from the *libs.data* file. The problem is that the library 1-100 database is very large, and when it tries to load the libs.data file, too, it totally wipes memory. The file is not corrupt and works fine as long as libs.data, which grows from month to month (and grew too large some time ago), is not on the same disk.

Libs.data is not too large for the library 101-x file.

The Compleat Old & New Testament on Your CMD Device

By Jeff Jones. The Compleat Bible was designed on a CMD system. It loads and runs fastest on such a system, not because it was designed for it, but

because the devices indeed are faster.

The Compleat Bible is the most disk-intensive program I've ever written. Couple that with a slow, limited capacity 1541 — even with JiffyDOS, and you're in for a wait, especially if you want to search all the way from Genesis to The Revelation.

The best place to run The Compleat Bible is in your RAMLink. Moving it there is a simple matter with the tools that came with your RAMLink. If you have a CMD hard drive or an FD-4000 with formatted ED disk, follow these same instructions.

First just use FCOPY to copy all of the Compleat Old Testament over to a 4-MB partition. If you don't have a 4-MB partition, stop. You can't do it. You'll need about 12,000 blocks free. That's not a typo.

Start with the first Old Testament disk and copy all of the files on disk one to your CMD device. Repeat the procedure for all of the sides of all of the disks in the Compleat Old Testament.

When you copy all of the files from the Old Testament, it's time to copy the New Testament. Choose all files, but when it comes time to copy, you must choose the Skip option so that no system files from the Old Testament are replaced with the files from the New Testament. Remember that the New Testament disks were written first, and the Old Testament fixed a few bugs.

Once again, this is crucial: You must not replace any system files of the Old Testament with system files from the New Testament. Copy the Old Testament first and always have the skip option on so that no file is replaced by FCOPY.

With the entire Bible in one partition, the Bible performs as intended, er, perhaps not by God, but certainly by me. You're never asked to insert another disk. The entire Bible is there for you to discover it. It's technically faster to have the Bible in the root directory than a subdirectory, but you won't notice much of a difference.

Hard And Fast REU Rules

By Robin Harbron. Want to know how to make use of a Ram Expansion Unit in your own programs? The following BASIC and assembly programs are examples of how to detect a REU, determine it's size and then store, retrieve or swap memory with it.

The REU is accessed through IO2, which is the memory locations from $DF00 to $DFFF. The REU actually has 7 registers, with some registers being multiple bytes, so all your PEEKing and POKEing is done in the addresses $DF00 to $DF0A (57088 to 57098).

How to detect a REU? If the REU is present, addresses $DF02 through $DF05 retain what is stored in them. I don't know of any other device with that property, so the following routines exploit this. In the BASIC version (which only uses one variable, just to be unobtrusive), x will equal 0 if there is no REU, and x=1 if there is one present.

```
10 rem detect
20
forx=2to5:poke57088+x,x:nextx
30 forx=5to2step-1
40 ifpeek(57088+x)<>xthenx=1
50 nextx
60 ifx=0thenprint"no reu"
```

In the assembly version (sys 16384 to start it from BASIC), the accumulator will be 0 if there is no REU, and 1 if there is one. PEEK (780) will tell you the contents of the accumulator if you call this routine from BASIC.

```
    *= $4000
    ; Detect REU
    ldx #2
loop1    txa
    sta $df00,x
    inx
    cpx #6
    bne loop1

    ldx #2
loop2    txa
    cmp $df00,x
    bne noreu
    inx
    cpx #6
    bne loop2

    lda #1
    rts

noreu    lda #0
    rts
```

Now, how to tell how big the REU is? This is my solution, although there seems to be no absolute best way. This routine takes a while to run, in the BASIC version (of course the assembly version takes no noticeable time at all). Simply speaking, the routine writes the bank number into the first byte of every bank (all 256 possible banks, which would be a 16-MB REU!). Banks that are not available seem to be shadows of the existing banks, so what you write into a shadow bank ends up overwriting what we had put in a previous bank. Then we just go through the banks again, and see how many consecutive, increasing values we can find. This number is the number of available banks.

This routine is also non-destructive. It reads and stores the original contents of the REU, and then puts them back when it's done. The routine requires a 257-byte buffer for this, which the variable s/label temp points to. After running the BASIC version, the variable A contains the number of banks available. For example, A=8 after running the program on my 512K REU. The ML version returns the number of banks in the accumulator.

```
10 rem size
15 s=49152
20 poke57090,0:poke57091,192
30 poke57092,0:poke57093,0
40 poke57095,1:poke57096,0
50 poke57098,0
60 forb=0to255:poke57094,b
63 pokes,b:poke57089,178
65 pokes+1+b,peek(s):nextb
70 b=0:o=0
80 poke57094,b:poke57089,177
90 n=peek(s)
100 ifn>otheno=n:b=b+1:goto80
110 a=b
120 forb=255to0step-
1:poke57094,b
130
pokes,peek(s+1+b):poke57089,176
140 nextb
150 printa
```

```
    *= $4000
temp   = $c000
    ;detect reu size

    lda #0
    sta $df04
    sta $df05
    sta $df08
    sta $df0a
    lda #1
    sta $df07

    lda #<temp
    sta $df02
    lda #>temp
    sta $df03

    ldx #0
loop1    stx $df06
    stx temp
    lda #178
    sta $df01
    lda temp
    sta temp+1,x
    inx
    bne loop1

    ldy #177
    ldx #0
    stx old
loop2    stx $df06
    sty $df01
```

```
    lda temp
    cmp old
    bcc next
    sta old
    inx
    bne loop2
next   stx size
    ldy #176
    ldx #255
loop3    stx $df06
    lda temp+1,x
    sta temp
    sty $df01
    dex
    cpx #255
    bne loop3
    lda size
    rts
old .byte 0
size   .byte 0
```

Now we get to the primary function of the REU: Storing and retrieving data. The following routine is a model for you to use. It either stashes or retrieves a screen to/from the REU. Here's the breakdown of the registers:
$DF02 & $DF03 (57090 & 57091) point to the base address in the computer's memory that we want to deal with. It's stored in standard lo/hi format. $400 is the address we stash in here to deal with the screen.
$DF04-$DF06 (57092-57094) point to the base address in the REU's memory that we want to use. It's stored in lo/hi/bank format. The bank number can be thought of as the "even higher" byte. We'll just store the info in location $0, bank 0.

$DF07-$DF08 (57095-57096) determine the length in bytes of the transfer. It's stored in lo/hi format as well. Storing a zero in this register causes the REU to transfer 64K at once, which is the limit for one move.
$DF0A allows you to fix either the C64 or REU memory address, to allow you to either write or read one particular memory location many times. If bit 7 is set, the C64 address is fixed, and if bit 6 is set, the REU address is fixed. This has many uses: You could fix the address in the REU (perhaps storing a 0 or 255 there) and then set the length to 8000 bytes. Then you could fill an entire bitmap screen with that byte in just 8000 cycles, by transferring from the REU to the C64! This is at least 4 times faster than doing it with the processor. Or how about sampling a particular location in memory? It's been used to examine the random number generator in the SID chip.
$DF01 (57089) actually executes the move. Setting bit 7 is necessary, to tell the REU to execute the move. The other bits tell the REU how to do that move: If bit 5 is set, the address and length registers are unchanged after the command is executed. This is useful if you are going to do many consecutive, identical transfers. If the bit is not set, the address registers will point to the address immediately after the last address accessed, and the length register will be set to 1. If bit 4 is set, the command will be executed immediately. If it is not set, the command will not be executed until you write to location $FF00. This sounds weird, but is useful if you want to do transfers around the IO/ROM areas of memory. This gives you a chance to set up the transfer, then switch IO out (for example), and then execute the transfer. A simple LDA $FF00 : STA $FF00 will do the job.

Bits 1 and 0 define the transfer type. 00 is for a C64 to REU move, 01 is for a REU to C64 move, 10 is to swap between REU and C64 memory (this takes twice as long, as it has double the work to do), and 11 is for a compare between the REU and C64. To use the compare feature, clear bit 5 of $DF00, and execute the compare (just like doing any other transfer, but no bytes are actually moved). Now, if bit 5 of $DF00 has been set, then a difference was found between the REU and C64 memory. Here's the BASIC version. Just make sure you set x according to the comments in line 60.

```
10 poke57090,0:poke57091,4:rem
c64
20poke57092,0:poke57093,0:
poke57094,0:rem REU
30 poke57095,232:poke57096,3:
rem length
40 poke57098,0:rem not fixed
addresses
50 poke57089,128+16+x
60 rem x=0 c64->reu, x=1 reu-
>c64, x=2 swap c64<->reu
```

In the assembly version, just set the labels c64 to the C64 base address, reu to the REU base address, bank to the bank number in the REU (I used 2, as my assembler makes use of 0 and 1), length to the number of bytes you want to transfer, and action equal to 0, 1, 2 or 3, corresponding to what type of transfer you'd like.

```
    *= $4000
    ;reu store/swap/get

c64 = $0400
reu = $00
bank   = $02
length   = 1000
action   = 0

    lda #<c64
    sta $df02
    lda #>c64
    sta $df03
    lda #<reu
    sta $df04
    lda #>reu
    sta $df05
    lda #bank
    sta $df06
    lda #<length
    sta $df07
    lda #>length
    sta $df08
    lda #0
    sta $df0a
    lda #144+action
    sta $df01
    rts
```

Stupid PET Tricks Part One

By Todd Elliott, with special thanks to Marko Mäkelä, msmakela@cc.hut.fi, for the corrections. Okay, okay! I'll admit that the heading isn't entirely original, but what the hey! If you're reading this sentence, then it has done its job, with sincere apologies to a nocturnal vaudeville comedian named Dave. I will attempt to illustrate a few down and dirty tricks to get the best out of your ML programming in Commodore computers, (Well, mostly for the C64/128.) if you'll forgive the pun on my part about the PET series computers. Now, onward!

1. Semi-documented JMP Instructions: BNE and BEQ Instructions.

Well, they have been documented as BRANCHing instructions, but their application for JMPing-type instructions usually is ignored. Let me illustrate: LDA CODE...JMP NEWROUTE. This code takes up five or six bytes, depending on the operand used in LDA CODE. It also takes six cycles to execute, due to the JMP instruction. But compare this with the LDA CODE...BNE NEWROUTE. This code shaves off one byte, an improvement over the old code. This replacement instruction, BNE works 99 percent of the time. However, if you do know that the zero flag will be set before the jump takes place, use BEQ instruction instead.

The main advantage is that it makes the code relocatable, saves bytes and is a little bit quicker. The main disadvantage is that clarity is lost. It is intended to BRANCH on a certain condition, instead of being used as a JMPing type instruction, making the resultant code harder to read, and it is limited to jumping only 127 bytes forward or 126 bytes backward. Ah, it would be nice to have

the BRA (BRanch Always) instruction...

2. CHAMELEON JSR INSTRUCTION.

It changes into a JMP Instruction! How many times when you JSR to a certain subroutine, and a condition resulted in that subroutine which requires exit somewhere else other than an RTS? You might as well as change it into a JMP in the first place. Or...you don't have to! First, an understanding of what a JSR instruction does: When the 65xx or 8502 microprocessor encounters a JSR instruction, it saves the incremented program counter on the stack in low byte, high byte format, and then JMPs to the subroutine. When the 65xx microprocessor encounters a RTS instruction, it fetches the last two bytes on the stack, and JMPs to the program counter, plus one. So, if you want to exit somewhere else, you can do this: PLA...PLA...JMP SOMEWHERE. The PLAs pull out the last two bytes that contained the program counter, and everything's all tidied up and neat, with the discarded addresses sent off to the PC orphanage in the sky.

Another devious use for the Chameleon JSR instruction would be the inline call. A good example is the PRIMM (PRint IMMediate) routine located at c128 Kernal jump entry $FF7D. The GEOS system also uses this method as an alternate to standard JSR routine calls. This is a good way to pass parameters between the main calling routine and the subroutine. For example:

```
JSR SubRoutine
.byte Xposition
.byte Yposition
.byte SpriteNum
[code continues...]
```

In this example, the main routine passed the x,y position coordinates of a certain sprite to SubRoutine. How does SubRoutine get the parameters? Easy! The JSR instruction stores the Program Counter into the stack, and you merely need to retrieve it like this:

```
SubRoutine =*
  pla  ;get the high byte of
program counter.
  sta zp+1;store it in a zero
page location or whatever.
  pla  ;get the low byte of
program counter.
  sta zp  ;store it in a zero
page location or whatever.
  inc zp  ;increment it by one
to get the true start address
of the
       ;parameters.
[code continues, massaging the
zp address...]
```

As you can see, the Chameleon JSR instruction can do a lot and is flexible enough even for the complicated subroutine structures that would make any computer science professor worth his/her salt to cringe and grasp his/her pocket protector.

3. MORE UNDOCUMENTED JMP INSTRUCTIONS: RTS AND BRK.

You think I'm jesting, huh? Nosiree! Read on! Remember the previous tip explaining how the RTS instruction works? You can create a program counter, minus one, and PHA the counter in a low byte/high byte format onto the stack, and then RTS! By doing this, you created a quasi-JMP, and the irony is that you use the RTS instruction to accomplish this. Pretty sneaky, huh? You could use this to end an inline call as shown earlier. Just keep track of the stored program counter at the zp address location as shown in the earlier example, modify it, (minus one, of course) and RTS to return back to the main routine code segment immediately following the parameters.

As for the BRK instruction, it does what it says- It interrupts the program and an interrupt handler at either $ffffa (when an NMI interrupt occurs at the same time) or $fffe processes the interrupt. Eventually, it will jump through the BReaK vector at $0316. You can modify the address in the BRK vector to point to a new address in a low byte/high byte format, and then all BRKs will go to this address, thereby creating a quasi-JMP to that address. Warning: since the NMI handler does not check the B flag, the BRK instruction could be lost in the actual event of an NMI interrupt happening at the same time.

The BRK technique is used in ML monitor programs. Both techniques can be used in your programs, but I'd advise against it. It consumes plenty of clock cycles, six for RTS and seven for BRK, and it pretty much makes the resulting code almost incomprehensible. Still, the flexibility accorded to the RTS instruction for self-modifying code and inline calls are usually worth the effort to code in the first place.

4. SELF-MODIFYING CODE.

You may have heard of it. Many ML tutorials only hint at it. Why? It contradicts the time honored principles of computer programming as fostered in a structured college curriculum, which emphasizes clean, clear and readable code above anything else and leaves no room for elegant 'trash' as this. But it's definitely worth your time to understand this little-known technique of ML programming. Example:

```
    LDA #$07
    LDY #$00
    STA ZP+1
    STY ZP
    LDA #$01
    LDY #$00
    LDX #$03
L1  STA (ZP),Y
    DEY
    BNE L1
    DEC ZP+1
    DEX
    BPL L1
    RTS
```

The preceding routine took 25 bytes to run, and an estimated time of 11,300+ clock cycles. Compare this with the following routine that uses self-modifying code:

```
    LDA #$07
    STA L1+2    ;stores the
high byte into location shown
at label L1.
    LDA #$01
    LDY #$00
    LDX #$03
L1  STA $FF00,Y ;The high byte
will be modified earlier when
first run,
          ;and will be
modified later by the decre-
menting routines.
    DEY
    BNE L1
    DEC L1+2    ;more modifying
hijinks, and decrements the
high byte
          ;stored in label L1.
    DEX
    BPL L1
    RTS
```

So far, the preceding routine took 24 bytes and an estimated time of 10,200+ clock cycles. The only apparent advantage of using self-modifying code is that of speed. You save a significant chunk of processing time this way to update the screen by 1,100 clock cycles. Also, using the absolute addressing mode helps the speed increase.

Also, please note that if you're programming in ML, you're ahead of the programming curve. Nowadays, Microsoft Corp. and other computer companies do not really sweat out their code or optimize their code. They just design the programs, and wait for the hardware to catch up and execute it at a satisfactory speed. We Commodore 8-bit users do not have that luxury, (SuperCPU owners notwithstanding) and resort to

these dirty tricks to get the most out of our machine. By doing so, these users will be better prepared for programming apps for the current computer platforms and sneer at fellow employees, "You use Visual C++? Ha, you're wimps! I assemble exclusively in 64-bit code for lunch!" While I do not have the statistics, I do firmly believe that the current crop of programmers probably cut their teeth poking away at their 8-bit computers and are far much better programmers because of the experience.

5. The Camouflaged BIT instruction.

What? An actual opcode fully decked out in army fatigues and ready to kick some serious coding mayhem? Does it wear camouflaged warpaint as to disguise himself among the eddy currents of the digital streams pulsating in the CBM 8-bit computers? Does it fire bullets of crisp code that would completely control the CBM 8-bit computer as to bring it to its knees? Well, not quite- the BIT instruction does have nice camouflaging qualities as shown in this example here:

```
LDA #$01
BIT $02A9
BIT $03A9
BIT $04A9
[etc. code continues...]
```

What is going on here? Well, this technique uses the BIT instruction as a smoke screen or a camouflage to hide the LDA instruction. If you look closely, each operand of the BIT instruction contains the opcode and operand for the LDA instruction, immediate addressing. More specifically, 'A9' is the hexadecimal equivalent of a LDA instruction, and the '02', '03' or the '04' are the hexadecimal equivalents of the LDA operand. Thus, the BIT $02A9 really means that it hides the LDA #$02 instruction.

Now, why? Why the need to hide the instructions? Maybe we are thinking of the BIT instruction in a mistaken fashion; it would make sense to think of BIT as a 'skipping' type of instruction. In this sense, it is a good way of reducing the code for many conditional branches. Only one result is selected and the rest skipped over. For example, the plain code, fully drawn out according to conventional, structured programming wisdom, would look like this: (In fact, even this code below can be optimized- just remove the JMP somewhere instructions, and it will still work correctly.)

```
  LDA zp; get the data that the
branching instructions would
rely on.
  BNE +
  LDA #$01
  JMP somewhere
+ CMP #$02
  BNE +
  LDA #$02
  JMP somewhere
+ CMP #$03
  BNE +
  LDA #$03
  JMP somewhere
+ CMP #$04
  BNE +
  LDA #$04
  JMP somewhere
+ [and so on... the branching
continues...]

somewhere =*
[the code continues with the
processing of the selected
result.]
```

But, with the BIT instruction, the code looks like this:

```
    LDA zp; get the data that
the branching instructions
would rely on.
    BEQ L1+1
    CMP #$02; arbitrary CoMPare
values. Could be anything you
desire.
    BEQ L2+1
    CMP #$03
    BEQ L3+1

    LDA #$01; again, arbitrary
values- just put in the desired
result.
L1  BIT $02A9
L2  BIT $03A9
L3  BIT $04A9
    [and so on... the code
continues...]
```

Once the routine selects the proper result (the LDA #$xx) of a branch condition, the subsequent BITs will not affect anything. In effect, you would choose the right result and skip the rest. Not only this routine is neater, it is shorter, saving thirteen bytes. With a lot of branch conditions with varying values, the benefits quickly pile up in terms of clarity and bytes saved.

I first saw this trick in the Oct.-Nov. 1990 issues of Gazette, written by none other than Jim Butterfield. Thank you for sharing this gem with the CBM programming community. Stephen Judd (judd@stratus.esam.nwu.edu) noted that you can substitute the BIT opcode with the compare opcodes using absolute addressing, i.e. CMP $02A9, and it will still work.

6. Use Zero Page Locations for your programs.

You aren't only limited to locations such as the cassette buffer to store programs or the 4K free area starting in $C000. You can 'plop' a small routine under around 142 bytes in zero page! When you enter the domain of ML, you forsake BASIC and its cozy environment. You can switch off BASIC and presto! You have around 142 contiguous free locations in zero page at your disposal. (From $02 onto $8F)

Why use it, you ask? Well, critical routines such as CHRGET have been placed in zero page to speed it up, and it is an excellent place if you are using a lot of variables. Supposedly if you have a critical screen updating routine, and you stored it in the 4K $C000 area, and you stored the same routine in zero page, the routine in zero page will win the race every time. The reason lies in the 65xx architecture. The microprocessor jumps to the program counter relative to zero page. The microprocessor will not have to go far if the routine/variable is already there in zero page, and executes it like lightning.

Last, before you even think about fiddling with zero page, it is common courtesy to save the zero page locations elsewhere, so that upon exit, you can restore these critical locations as to prevent the BASIC OS from locking up the computer. One caveat- I did try this, and ran the program in a C64S emulator shareware version, and it crashed. One more reason to stick with the classic c64! Okay, it was an older version, v1.1, I believe. The zero page technique should work correctly with the newer crop of c64 emulators.

Allegedly, the beguiling ideas about science quoted here were gleaned from essays, exams, and classroom discussions, and most were from 5th and 6th graders:

One horsepower is the amount of energy it takes to drag a horse 500 feet in one second.

You can listen to thunder after lightning and tell how close you came to getting hit. If you don't hear it, you got hit, so never mind.

Talc is found on rocks and on babies.

The law of gravity says no fair jumping up without coming back down.

When they broke open molecules, they found they were only stuffed with atoms. But when they broke open atoms, they found them stuffed with explosions.

When people run around and around in circles, we say they are crazy.

When planets do it, we say they are orbiting.

Rainbows are just to look at, not to really understand.

Someday we may discover how to make magnets that can point in any direction.

South America has cold summers and hot winters, but somehow they still manage.

Most books now say our sun is a star. But it still knows how to change back into a sun in the daytime.

Water freezes at 32 degrees and boils at 212 degrees. There are 180 degrees between freezing and boiling because there are 180 degrees between north and south.

A vibration is a motion that cannot make up its mind which way it wants to go.

There are 26 vitamins in all, but some of the letters are yet to be discovered. Finding them all means living forever.

There is a tremendous weight pushing down on the center of the Earth because of so much population stomping around up there these days.

Many dead animals in the past changed to fossils while others preferred to be oil.

Vacuums are nothings. We only mention them to let them know we know they're there.

Some oxygen molecules help fires burn while others help make water, so sometimes it's brother against brother.

Some people can tell what time it is by looking at the sun. But I have never been able to make out the numbers.

To most people solutions mean finding the answers. But to chemists solutions are things that are still all mixed up.

Clouds are high-flying fogs.

I am not sure how clouds get formed. But the clouds know how to do it, and that is the important thing.

Clouds just keep circling the earth around and around. And around. There is not much else to do.

Water vapor gets together in a cloud. When it is big enough to be called a drop, it does.

Humidity is the experience of looking for air and finding water.

We keep track of the humidity in the air so we won't drown when we breathe.

Rain is often known as soft water, oppositely known as hail.

Rain is saved up in cloud banks.

In some rocks you can find the fossil footprints of fishes.

A blizzard is when it snows sideways.

A hurricane is a breeze of a bigly size.

Thunder is a rich source of loudness.

Isotherms and isobars are even more important than their names sound.

LOADSTAR LETTER #47

J&F Publishing • 606 Common Street • Shreveport LA 71101

- Commodore News
- Commodore Views

Bulk Rate
U.S Postage PAID
Shreveport LA
PERMIT #85